Amendments to the claims,
Listing of all claims pursuant to 37 CFR 1.121(c)

This listing of claims will replace all prior versions, and listings, of claims in the application:

What is claimed is:

1. (Currently amended) A method for providing access to data in a multi-threaded computing system, the method comprising:

providing a cache containing pages of data in memory of the multi-threaded computing system;

associating a latch with each page in the cache to regulate access to the page, the latch allowing multiple threads to share access to the page for read operations and a single thread to obtain exclusive access to the page for write operations;

in response to a request from a first thread to read a particular page, determining whether the particular page is in the cache without blocking access by other threads to pages in the cache, wherein said determining step includes determining whether the particular page is in the cache without acquiring a mutual exclusion object (mutex) controlling access to pages in the cache;

if the particular page is in the cache, attempting to obtain the latch for purposes of reading the particular page; and

allowing the first thread to read the particular page unless a second thread has latched the particular page on an exclusive basis.

2. (Original) The method of claim 1, wherein the pages of data comprise database file pages.

3. (Original) The method of claim 1, wherein said providing step includes organizing the pages of data in an array.

4. (Original) The method of claim 1, wherein said providing step includes providing an index facilitating access to pages in the cache.

5. (Original) The method of claim 4, wherein said step of providing an index includes providing an index based upon a unique identifier assigned to each page in the cache.

6. (Original) The method of claim 1, wherein said providing step includes maintaining state information for each page in the cache.

7. (Original) The method of claim 6, wherein said step of maintaining state information for each page includes assigning a unique identifier to each page.

8. (Original) The method of claim 7, wherein said unique identifier comprises a page name.

9. (Original) The method of claim 7, wherein said determining step includes determining whether the particular page is in cache based, at least in part, upon said unique identifier.

10. (Original) The method of claim 6, wherein the state information maintained for each page includes the latch for regulating access to the page.

11. (Canceled)

12. (Original) The method of claim 1, wherein said allowing step includes allowing the first thread to read the particular page concurrently with another thread reading the particular page.

13. (Original) The method of claim 1, further comprising:

if the particular page is not in the cache, attempting to install a cache entry for the particular page.

14. (Original) The method of claim 13, wherein said attempting to install step

includes obtaining an unused cache entry.

15. (Original) The method of claim 14, wherein said attempting to install step includes latching the unused cache entry on an exclusive basis.

16. (Currently amended) The method of claim 13, wherein said attempting to install step includes acquiring a mutual exclusion object (mutex) controlling access to a cache chain and attempting to install ~~the~~ an unused cache entry in the cache chain.

17. (Original) The method of claim 1, further comprising:

if a second thread has latched the particular page on an exclusive basis, attempting to prevent reuse of the particular page.

18. (Original) The method of claim 17, wherein said attempting to prevent reuse step includes determining if the particular page is in the cache.

19. (Original) The method of claim 17, wherein said attempting to prevent reuse step includes incrementing an indicator associated with the particular page so as to indicate the first thread is waiting for access to the particular page.

20. (Original) The method of claim 1, further comprising:

maintaining a list of reusable pages representing pages in the cache that are available for reuse.

21. (Original) The method of claim 20, wherein the list of reusable pages is structured as a double ended queue.

22. (Original) The method of claim 21, further comprising:

maintaining a gap between ends of the double ended queue.

23. (Original) The method of claim 21, further comprising:

identifying a page in the cache that can be reused; and

adding the identified page to one end of the double ended queue.

24. (Original) The method of claim 23, wherein said step of adding the identified page includes adding the identified page using thread-unsafe operations so as to provide improved system performance.

25. (Original) The method of claim 23, further comprising:

removing a page from the other end of the double ended queue when a page is needed.

26. (Original) The method of claim 25, wherein said step of removing the page includes removing the page using thread-unsafe operations so as to provide improved system performance.

27. (Original) A computer-readable medium having processor-executable instructions for performing the method of claim 1.

28. (Currently amended) The method of claim 1, further comprising:

~~A downloadable~~ downloading a set of processor-executable instructions for performing the method of claim 1.

29. (Currently amended) A system for providing access to data in a multi-threaded computing system, the system comprising:

a cache containing pages of data in memory of the multi-threaded computing system;

a latch associated with each page in the cache to regulate access to the page, the latch allowing multiple threads to share access to the page for read operations and a single thread to obtain exclusive access to the page for write operations; and

a cache manager for receiving a request for reading a particular page from a first thread, determining whether the particular page is in the cache without blocking access

by other threads to pages in the cache, attempting to obtain the latch for reading the particular page if the particular page is in the cache; and allowing the first thread to read the particular page unless another thread has latched the particular page on an exclusive basis, wherein the cache manager determines whether the particular page is in the cache without acquiring a mutual exclusion object (mutex) regulating access to pages in the cache.

30. (Original) The system of claim 29, wherein the pages of data comprise database file pages.

31. (Original) The system of claim 29, wherein the cache includes an array holding the pages of data.

32. (Original) The system of claim 29, wherein the cache includes an index facilitating access to the pages in the cache.

33. (Original) The system of claim 32, wherein said index is based upon a unique identifier assigned to each page in the cache.

34. (Original) The system of claim 29, wherein the cache includes state information for each page in the cache.

35. (Original) The system of claim 34, wherein the state information includes a unique identifier for each page.

36. (Original) The system of claim 35, wherein said unique identifier comprises a page name.

37. (Original) The system of claim 35, wherein the cache manager determines whether the particular page is in cache based, at least in part, upon said unique identifier.

38. (Canceled)

39. (Original) The system of claim 29, wherein the cache manager allows the first thread to read the particular page concurrently with another thread reading the particular page.

40. (Original) The system of claim 29, wherein the cache manager attempts to install a cache entry for the particular page if the particular page is not found in cache.

41. (Original) The system of claim 29, wherein the cache manager attempts to prevent reuse of the particular page if another thread has latched the page on an exclusive basis.

42. (Original) The system of claim 29, further comprising:
a list of reusable pages representing pages in the cache that are available for reuse.

43. (Original) The system of claim 42, wherein the list of reusable pages is structured as a double ended queue.

44. (Original) The system of claim 43, wherein the cache manager identifies a page in the cache that can be reused and adds the identified page to one end of the double ended queue.

45. (Original) The system of claim 44, wherein cache manager adds the identified page using thread-unsafe operations so as to provide improved system performance.

46. (Original) The system of claim 44, wherein the cache manager removes a page from the other end of the double ended queue when a page is needed.

47. (Original) The system of claim 46, wherein the cache manager removes the page using thread-unsafe operations so as to provide improved system performance.